Spine Consortium

- Juha Kiviluoma, VTT
- Hannele Holttinen, VTT
- Erkka Rinne, VTT
- Tomi J. Lindroos, VTT
- Pekka T. Savolainen, VTT
- Janne Keränen, VTT
- Antti Lehtilä, VTT
- Laura Sokka, VTT
- Toni Lastusilta, VTT
- Irina Granfors, VTT
- Timo Kyntäjä, VTT
- Terence O'Donnell, UCD
- Lucy Cradden, UCD
- Jonathan Ruddy, UCD
- Mark O'Malley, UCD / Energy Reform Ltd.
- Erik Delarue, KUL
- Kris Poncelet, KUL
- Maren Ihlemann, KUL
- Steffen Kaminski, KUL
- Lennart Söder, KTH
- Mikael Amelin, KTH
- Lars Nordström, KTH
- Manuel Marin, KTH
- Jon Olauson, KTH
- Jody Dillon, Energy Reform Ltd.

www.spine-model.org
- Open-source energy modeling toolbox
- Open-source generic energy-system optimization model
Spine Toolbox: goals

Open
• Github (online repository)
• Use open source software: Python

Flexible
• User friendly model development:
  • Direct link between data and model (easily add new parameters, entities, etc.)
• Ability to plug in different (types of) energy system models:
  • Optimization, agent-based, etc.
  • Julia, GAMS, Python
• Facilitate soft-linking between different models

Practical
• Graphical user interface (GUI)
• Convenient handling of input data:
  • Connection to different types of data sources (SQL, excel, manual entry, etc.)
  • Incorporate data processing scripts
• Facilitate scenario creation and management
• Viewing functionalities
Spine Toolbox: Main View

- GUI visualizing data stores, tools (models/scripts), views (viewing scripts)
- Drawing of connections to link data stores to tools or views
- API allows connecting to different types of databases (e.g., MySQL, SQLite, etc.)
Spine Toolbox: goals

Open

- Github/Gitlab
- Use open source software: Python

Flexible

- User friendly model development:
  - Direct link between data and model (easily add new parameters, entities, etc.)
- Ability to plug in different (types of) energy system models:
  - Optimization, agent-based, etc.
  - Julia, GAMS, Python
- Facilitate soft-linking between different models

Practical

- GUI
- Convenient handling of input data:
  - Connection to different types of data sources (SQL, excel, manual entry, etc.)
  - Incorporate data processing scripts
- Facilitate scenario creation and management
- Viewing functionalities

- Generic data structure
- API for data access
Spine project - scope

- Open-source energy modeling toolbox
- Open-source generic energy-system optimization model
Spine’s generic data store: toolbox tree view

- **ObjectClass**: direction, commoditygroup, unitgroup, connection, unittemplate, unit, CoalPlant, GasPlant, CHPPlant, ImportGas, ImportCoal
- **Object**: commodity, Heat, Gas, Electricity, Coal, commoditygroup_commodity
- **RelationshipClass**: connection_node_node
- **Relationship**: connection_node_node CoalPlant_Coal, ImportCoal_Coal

**Parameter**

- **Object parameters**
  - unit: CoalPlant, available_factor: 1
  - unit: GasPlant, available_factor: 1
  - unit: CHPPlant, available_factor: 1
  - node: Leuven
  - node: BrusselsElectricity, demand: 1
  - node: AntwerpElectricity, demand: 1
  - node: BelgiumHeat, demand: 1
  - connection: EL1

- **Relationship parameters**
  - connection_node_node: EL1, LeuvenEl
  - connection_node_node: EL1, LeuvenEl
  - connection_node_node: EL1, Antwerp
  - connection_node_node: EL1, Antwerp
  - connection_node_node: EL1, LeuvenEl
  - connection_node_node: EL2, LeuvenEl
  - connection_node_node: EL2, BrusselsEl
  - connection_node_node: EL2, BrusselsEl
Spine project - scope

- Open-source energy modeling toolbox
- Open-source generic energy-system optimization model
Spine database API: convenience functions

Problem specific data structure:
- Addition of new parameter or new entity requires changing:
  - data structure
  - interface

Problem independent data structure:
- Parameters/entities entered in the toolbox can directly be accessed in the model
Example: Spine database API & convenience functions

- Convenience functions:
  - Use database structure to **automatically** generate functions to access objects/relationships/parameters

```plaintext
function max_flow(m::Model, flow)
    @constraint(
        m,
        [ u in unit(),
        c in commodity(),
        max_flow(unit=u, commodity=c) != nothing,
        flow[c, u, t] <= max_flow(unit = u, commodity = c)
    ),
end
```

- Functions to access set of all object of a certain object class
- Functions to access parameter values
Open-source energy modeling toolbox

Open-source generic energy-system optimization model
Spine Model: design goals and approaches

Open
- Github
- Use open-source software: Julia

Flexible
- One model generator for wide range of applications:
  - Long-term energy system optimization ~TIMES
  - Detailed UC models ~PLEXO
  - Hydro scheduling
  - Heat system optimization with building heating physics
  - etc.
- Easy addition of new parameters/entities/constraints

Fast
- Julia
- Efficient formulations
- Parallelization/decomposition techniques

- Generically defined constraints
  - Problem independent formulation
  - Commodity agnostic
  - Flexible geographical structure
  - Flexible temporal structure
  - Specific constraints
  - “user-constraints”

- Convenience functions
Project status

- Spine database API: git@github.com:Spine-project/Spine-Database-API.git
- Spine Toolbox: git@github.com:Spine-project/Spine-Toolbox.git
- Spine Model: git@github.com:Spine-project/Spine-Model.git
www.spine-model.org

spine_info@vtt.fi

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N. 774629.